ASSIGNMENT 3

Q 1.

Find and replace strings. The following exercise involves replacing characters in a specified string value. The first input field will indicate which character string will be replaced, and the second input field will indicate which characters will replace them once the button is clicked.

Use the HTML below as a template, and add the JavaScript needed to complete the task:

```html
<!doctype html>
<html>

<head>
    <title>Complete JavaScript Course</title>
</head>

<body>
    <div id="output">Complete JavaScript Course</div>
    Search for:
    <input id="sText" type="text">
    <br> Replace with:
    <input id="rText" type="text">
    <br>
    <button>Replace</button>
    <script>
    </script>
</body>

</html>
```

Take the following steps:

1. Select each of the three page elements using JavaScript and assign the element objects as variables so that they can be easily referenced in your code.

2. Add an event listener to the button to invoke a function when clicked.

3. Create a function named lookup() that will find and replace the text in the output element. Assign the output element's text content to a variable named s, and then assign the value of the input we are replacing to another variable named rt.

4. Create a new regex with the value of the first input field, which will allow you to replace the text. Using the regex, check for a match with the match() method. Wrap this with a condition that will execute a block of code if matches are found.

5. If the match is found, use replace() to set the new value.

6. Update the output area with the newly created and updated text output.

Q 2.
Create an application that uses JavaScript to check whether the string value of an input is a validly formatted email using regex. Look at the following template:

```html
<!doctype html>
<html>

<head>
    <title>JavaScript Course</title>
</head>

<body>
    <div class="output"></div>
    <input type="text" placeholder="Enter Email">
    <button>Check</button>
    <script>
    </script>
</body>

</html>
```

Take the following steps:
1. Use the above template code to start creating your application. Within the JavaScript code, select the input, output, and button elements from the page as JavaScript objects.
2. Add an event listener to the button to run a block of code when clicked that will get the current value in the input field. Create a blank response value that will populate the output div element contents.
3. Add a test with the string value from the input field and the expression for email format. If the test result is false, update the response output to say Invalid Email and change the output color to red.
4. If the condition of the test returns true, add a response that confirms the email format is correct and change the text color of output to green.
5. Output the response value into the output element.

Q3.
Email extractor
Use the following HTML as a starter template and add the JavaScript code to make an email extractor function:

```html
<!doctype html>
<html>

<head>
```

```html
    <title>Complete JavaScript Course</title>
</head>

<body>
    <textarea name="txtarea" rows=2 cols=50></textarea> <button>Get
        Emails</button>
    <textarea name="txtarea2" rows=2 cols=50></textarea>
    <script>
    </script>
</body>

</html>
```

Take the following steps:
1. In JavaScript, select both text areas and the button and set them as JavaScript objects.
2. Add an event listener to the button that will invoke a function that gets the content of the first textarea and filters it to only accept email addresses.
3. Within the extracting function, get the content of the first input field. Using match(), return an array of the email addresses that were matched from within the content from the first textarea.
4. To remove any duplicates, create a separate array that will hold only unique values.
5. Loop through all the email addresses found and check whether each one is already in the holder array, and if not, add it.
6. Using the join() array method, you can now join together the results of the email addresses found within the content and output it into the second textarea.

Q4.
Fetching Data from a JSON File using AJAX Write a JavaScript function that fetches data from a JSON file using AJAX and displays it on the webpage.
The function should perform the following tasks:
1. Make an AJAX request to fetch the data from a JSON file.
2. Parse the JSON response.
3. Display the data on the webpage.

Q5.
Updating Data to a JSON File using AJAX Create a web page with a form that allows users to update data in a JSON file using AJAX. The form should have fields to enter the data to be updated and a button to submit the form.
The JavaScript code should perform the following tasks:
1. Capture the form submission event.
2. Retrieve the data entered in the form fields.
3. Make an AJAX request to update the JSON file with the new data.
4. Display a success message if the update is successful.
Q6.

Filtering Data from a JSON API using AJAX Write a JavaScript function that fetches data from a JSON API using AJAX and filters the results based on user input.
The function should perform the following tasks:
1. Retrieve user input for filtering criteria (e.g., a specific property value).
2. Make an AJAX request to fetch data from the JSON API.
3. Parse the JSON response.
4. Filter the data based on the user input.
5. Display the filtered results on the webpage.
Note: Remember to handle any errors that may occur during the AJAX requests and provide appropriate feedback to the user.

Q7.
Fetching Data from an XML File using AJAX Write a JavaScript function that fetches data from an XML file using AJAX and displays it on the webpage.
The function should perform the following tasks:

1. **Make an AJAX request to fetch the XML data.**
2. **Parse the XML response.**
3. **Extract the required data from the XML and display it on the webpage.**

Q8.
Updating Data to an XML File using AJAX Create a web page with a form that allows users to update data in an XML file using AJAX. The form should have fields to enter the data to be updated and a button to submit the form.
The JavaScript code should perform the following tasks:

1. **Capture the form submission event.**
2. **Retrieve the data entered in the form fields.**
3. **Make an AJAX request to update the XML file with the new data.**
4. **Display a success message if the update is successful.**

Q9.
Filtering Data from an XML API using AJAX Write a JavaScript function that fetches data from an XML API using AJAX and filters the results based on user input.
The function should perform the following tasks:

1. **Retrieve user input for filtering criteria (e.g., a specific tag or attribute value).**
2. **Make an AJAX request to fetch data from the XML API.**
3. **Parse the XML response.**
4. **Filter the data based on the user input.**
5. **Display the filtered results on the webpage.**

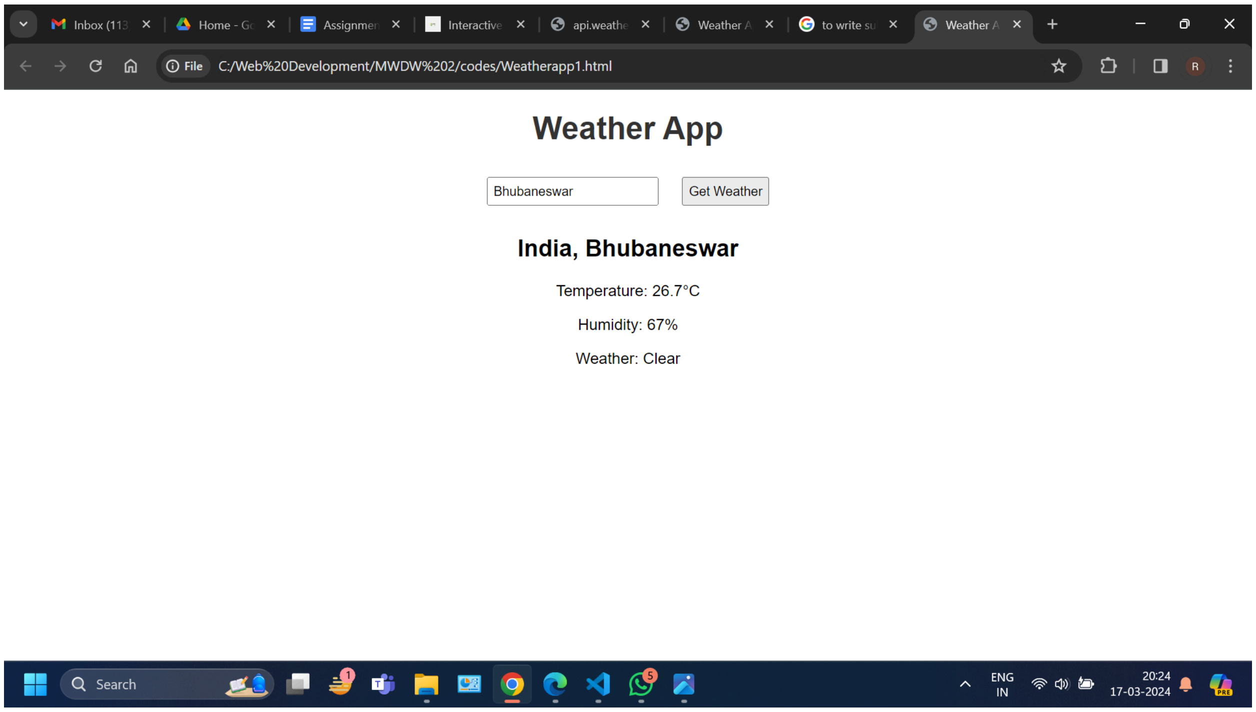
Note: Remember to handle any errors that may occur during the AJAX requests and provide appropriate feedback to the user.

Q10.

Create a Weather App Webpage Using Html and AJAX Write a JavaScript function that fetches data from an Weather API and displays it on the Html Page.
The project should perform the following tasks:

1. **Use an Input area with Button to enter the city of choice.**
2. **Display The Weather information**
    a. **Wind_mph**
    b. **Wind_kph**
    c. **Wind_degree**
    d. **Wind_direction**
    e. **Pressure_in**
    f. **Humidity**
    g. **Temperature**
3. **Display the Weather Condition information**

Here is a sample Output



Weather App

Bhubaneswar

Get Weather

India, Bhubaneswar

Temperature: 26.7°C

Humidity: 67%

Weather: Clear

Note: Use this website for free Weather API https://www.weatherapi.com